НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Вебдодаток для ведення журналу успішності школярів

Виконав

студент IV курсу, групи КВ-61
(шифр групи)

Рекеда Володимир Валерійович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц. каф. СПіСКС, к. т. н., доцент Клятченко Я.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Рекеди Володимира Валерійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту «Вебдодаток для ведення журналу успішності школярів», керівник проєкту доц. каф. СПіСКС, к. т. н., доцент Клятченко Я.М.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 2020 р. №_____

2. Термін подання студентом проєку

3. Вихідні дані до проєкту:

- веб-орієнтований додаток для ведення журналу успішності школярів.

4. Зміст пояснювальної записки

- аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
- аналіз технологій, які використовуються при розробці вебдодатків;
- опис розробленого вебдодатку для ведення журналу успішності школярів.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- структура бази даних вебдодатку (схема структурна);
- архітектура вебдодатку (схема структурна);
- алгоритм реєстрації нового користувача (схема алгоритму);
- взаємодія модулів вебдодатку (схема структурна).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "30" жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	15.02.2020	
2.	Розроблення та узгодження технічного завдання	10.03.2020	
3.	Аналіз існуючих рішень	20.03.2020	
4.	Розробка архітектури вебдодатку	02.04.2020	
5.	Розробка серверної частини вебдодатку	07.04.2020	
6.	Розробка клієнтської частини вебдодатку	17.04.2020	
7.	Тестування веб-додатку	27.04.2020	
8.	Підготовка текстової частини дипломного проєкту	05.05.2020	
9.	Підготовка матеріалів графічної частини проєкту	12.05.2020	
10.	Оформлення документації дипломного проєкту	14.05.2020	

Студент _____ Володимир РЕКЕДА
(підпис)

Керівник проєкту _____ Ярослав КЛЯТЧЕНКО
(підпис)

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.002 ТЗ	Вебдодаток для ведення	4		
			журналу успішності			
			школярів			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Вебдодаток для ведення	1		
			журналу успішності			
			школярів			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045440.004 ПЗ	Вебдодаток для ведення	51		
			журналу успішності			
			школярів			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Вебдодаток для ведення	1		
			журналу успішності			
			школярів			
			Структура бази даних			
			вебдодатку			
			Схема структурна			

Зм	Лист	№ докум.	Підп	Дата	ІАЛЦ.045440.001 ОА		
Розроб.		Рекеда В.В.			Вебдодаток для ведення журналу успішності школярів *Опис альбому*	Літ.	Лист 1 / Листів 2
Перев.		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-61	
Н. контр.		Клятченко Я.М.					
Затв.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.006 Д2	Вебдодаток для ведення	1		
			журналу успішності			
			школярів			
			Архітектура вебдодатку			
			Схема структурна			
	А4	ІАЛЦ.045440.007 Д3	Вебдодаток для ведення	1		
			журналу успішності			
			Алгоритм реєстрації			
			нового користувача			
			Схема алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Вебдодаток для ведення	1		
			журналу успішності			
			Взаємодія модулів			
			вебдодатку			
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Вебдодаток для ведення журналу успішності школярів».

Галузь застосування: навчальні заклади.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення вебдодатку для ведення журналу успішності школярів

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з операційною з операційними ситемами: Windows, Linux;
- сумісність з веб-браузерами: Google Chrome, Opera;
- можливість вносити оцінки для вчителя;

- можливість переглядати оцінки для учня;
- простий та інтуїтивно-зрозумідий інтерфейс;
- система авторизації з поділом користувачі на ролі.

5.2. Вимоги до апаратного забезпечення

- Оперативна пам’ять: 4 Гб;
- Наявність доступу до глобальної мережі Internet.

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Linux;
- Веб-браузер .

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.02.2020
2.	Розроблення та узгодження технічного завдання	10.03.2020
3.	Аналіз існуючих рішень	20.03.2020
4.	Розробка архітектури вебдодатку	02.04.2020
5.	Розробка серверної частини вебдодатку	07.04.2020
6.	Розробка клієнтської частини вебдодатку	17.04.2020
7.	Тестування вебдодатку	27.04.2020
8.	Підготовка текстової частини дипломного проєкту	05.05.2020
9.	Підготовка матеріалів графічної частини проєкту	12.05.2020
10.	Оформлення документації дипломного проєкту	14.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.004 ПЗ	Вебдодаток для ведення	51		
			журналу успішності			
			школярів			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Вебдодаток для ведення	1		
			журналу успішності			
			школярів			
			Структура бази даних			
			вебдодатку			
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Вебдодаток для ведення	1		
			журналу успішності			
			школярів			
			Архітектура вебдодатку			
			Схема структурна			
	А4	ІАЛЦ.045440.007 Д3	Вебдодаток для ведення	1		
			журналу успішності			
			Алгоритм реєстрації			
			нового користувача			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Рекеда В.В.			Вебдодаток для ведення журналу успішності школярів Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Клятченко Я.М.					
Консульт.						**КПІ ім. Ігоря Сікорського, ФПМ, КВ-61**	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Схема алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Вебдодаток для ведення	1		
			журналу успішності			
			Взаємодія модулів			
			вебдодатку			
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – база даних.

ОС – Операційна система.

СКБД (Система керування базами даних) – сукупність програмних засобів загального або спеціального призначення, які забезпечують керування створенням і використанням баз даних.

ACID (Atomicity, Consistency, Isolation, Durability) – атомарність, узгодженість, ізольованість, довговічність - набір властивостей БД, які необхідні для надійної роботи транзакцій.

CSS (Cascading Style Sheets) – формальна мова опису зовнішнього вигляду документа, написаного з використання мови розмітки.

HHTP (HyperText Transfer Protocol) – протокол передачі даних прикладного рівня, є основою передачі даних в мережі Інтернет.

JSON (JavaScript Object Notation) – текстовий формат представлення структурованих даних, заснований на синтаксисі об’єкта в мові JavaScript.

MVC (Model-View-Controller) – архітектурний шаблон, який застосовується для розробки додатків.

MVP (Model-View-Presenter) - архітектурний шаблон, який застосовується для розробки додатків, виник на базі MVC.

MVVM (Model-View-ViewModel) - архітектурний шаблон, який застосовується для розробки додатків, виник як модифікація MVP.

RDBMS (Relational Database Management System) - сукупність програмних засобів загального або спеціального призначення, які забезпечують керування створенням і використанням реляційних баз даних.

SQL (Structured Query Language) – мова структурованих запитів, використовується для керування даними, які містять реляційній системі управління базами даних.

T-SQL – (Transact - Structured Query Language) – процедурне розширення мови SQL, створене компанією Microsoft (для Microsoft SQL Server) і Sybase (для Sybase ASE).

XML (Extensible Markup Languag) - мова розмітки, яка визначає набір правил для кодування документів, запропонована консорціумом World Wide Web Consortium (W3C).

ВСТУП

В XXI ст. життя людини стає все тісніше пов’язаним з Інтернетом. Його популярність різко зростає та все дедалі більше сфер людського життя діджиталізується. Сфери, що пов’язані з документообігом, також почали використовувати Інтернет з метою розвитку систем та полегшення проведення стандартних операцій. Успішним прикладом такої діджиталізації є створення українського додатку «Дія», за допомогою якого можна використовувати електронні версії документів, таких як паспорт, водійське посвідчення, студентський квиток та інші. Даний додаток лише нещодавно почав функціонувати, проте, однозначно користується популярністю, а його засновники у свою чергу вдосконалюють та розвивають свою розробку.

Використання електронних версій документів у наш час, може суттєво спростити життя людини, адже всі документи завжди будуть зібрані в одному місці, а для доступу до них буде необхідно мати лише пристрій з доступом до мережі Інтернет. Електронний документообіг зменшує час на обробку документів, виключає втрату документів, зберігає навколишнє середовище, за рахунок економії паперу та має багато інших переваг. Окрім того, всі звіти та реєстри створюються автоматично, у результаті чого, робота з документами відбувається якісніше та оперативніше та призводить до оптимізації багатьох процесів.

Подібні системи також можуть знайти своє застосування у сфері освіти. Процес оптимізації ведення документів у школах, університетах та інших навчальних закладах може значно покращити системи, якими користуються на даний момент. На щастя, у більшості університетів вже існують подібні системи, які створювалися окремо, наприклад, електронна система «Кампус» у НТТУ «КПІ ім. Ігоря Сікорського». Проте, значно гірша ситуація у організації ведення документів у школах. На жаль, вони мають менше можливостей для створення подібних систем, а тому працюють згідно застарілих методів. Саме

тому було би доцільно розробити єдину систему або ж певне шаблонне рішення.

Дана система може складатися з кількох компонентів, наприклад, таких як електронний щоденник, електронний журнал, електронна бібліотека та інші. У свою чергу це полегшить життя вчителів, учнів та батьків, допоможе контролювати навчальний процес та значно зекономить час.

У даному бакалаврському проєкті буде акцентовано увагу саме на електронний журнал. Створення такого компоненту може дуже спростити формування звітності з успішності навчального процесу за рахунок автоматизації цього процесу. Даний компонент буде представлено у форматі веб-додатку, адже такий формат програмного забезпечення є досить зручним завдяки його адаптивності.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Загальний опис проблеми

Нині не існує єдиної системи для шкіл, але її створення було б досить доречним. Дана система має забезпечувати ведення журналу успішності в електронному вигляді. Крім того, в ній має бути два режими доступу: для вчителів виставлення оцінок, для учнів та їх батьків – для перегляду успішності учня. Така система б задовольняла вимогам анонімності, щодо успішності конкретного учня – лише вчитель, учень та його батьки знають про отримані бали.

Завдяки існуванню подібної системи батьки завжди матимуть доступ до інформації про успішність їх дитини і зможуть і контролювати. Крім того, до даної системи можна додати і деякі інші елементи щоденника. Наприклад, інформацію про розклад, задані домашні завдання, певні оголошення, посилання на веб-ресурси з книгами та посібниками.

Також дана система була б дуже корисною при виникненні певних форс-мажорних ситуацій, коли похід до реальної школи є неможливим, наприклад, під час карантину.

Розробку даної системи доцільно проводу у вигляді веб-додатку. Причини для цього такі:

- Для роботи веб-додатку користувач не потрібно встановлювати додаткове програмне забезпечення, необхідна лише наявність веб-браузера;
- Розробка веб-додатку є простіша та швидша, ніж розробка окремого віконного додатку;

- На функціонування веб-додатку не впливає ОС, яка встановлена у користувача.

1.2. Аналіз існуючих рішень

На даний час існує кілька українських платформ, які надають можливості ведення журналів успішності.

Як один з прикладів можна назвати платформу E-Schools [1] (рис 1.1). Даний сервіс є безкоштовним для шкіл. На даній платформі є кілька сервісів, які існують як окремі веб-додатки, до них відносяться: електронний щоденник, чат платформа для спілкування всередині шкіл, також є можливість створити сайт школи. Також на даній платформі існує поділ користувачів на вчителів та батьків/учнів.

Наповненням інформації в даному випадку має займатись модератор школи, тобто дана платформа не є централізованою і може змінюватись під потреби кожної окремої школи.

З недоліків можна вказати перевантаженість веб-додатку, що робить його складним для сприйняття.

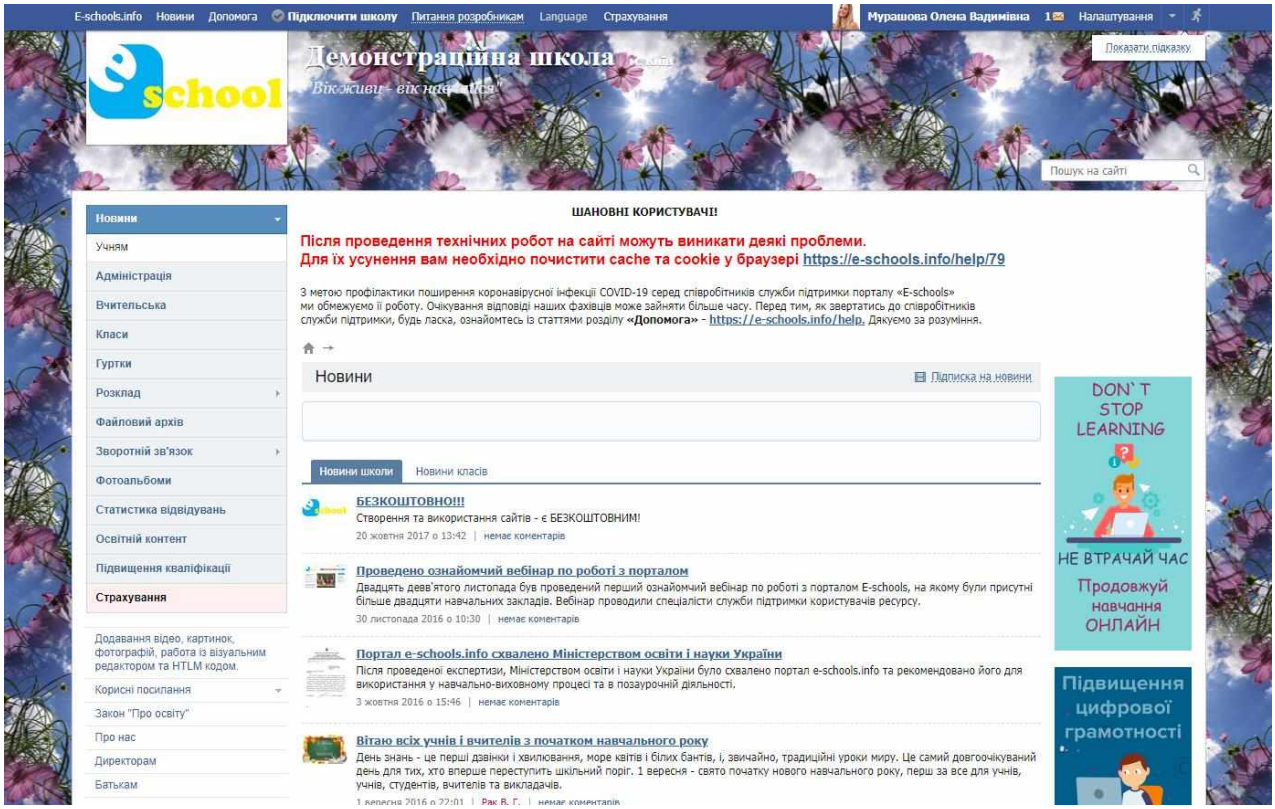

Рисунок 1.1 – Платформа E-Schools

Як інший приклад можна назвати платформу shodennik.ua [2] (рис 1.2). Платформа створена українськими розробниками та є безкоштовною. Дана платформа формує електронне середовище для учнів/їх батьків та вчителів.

Особливістю даної платформи є наявність частин, які роблять її схожою з соціальною мережею, наприклад, всі користувачі мають власні профілі, можуть додавати один одного у друзі, існують повідомлення, групи, блоги та стрічка новин.

Саме ця особливість є одночасно перевагою та недоліком даної платформи. З одної сторони, рішення поєднати шкільне спілкування однокласників та навчальну частину школи є досить цікавим, адже таким чином створюється єдиний простір, який дуже схожий на навчання в офлайні. З іншої сторони, нагромадження груп, повідомлень, дописів у стрічці новин може відволікати від самого навчання. Наприклад, учень зайшов дізнатись своє домашнє завдання з метою виконати його, але його зацікавила переписка та

перегляд постів у стрічці новин, які затягнулись на кілька годин. Крім того, нині існує досить велика кількість соціальних мереж та месенджерів, які є більш комфортними та мають більш обширний функціонал.

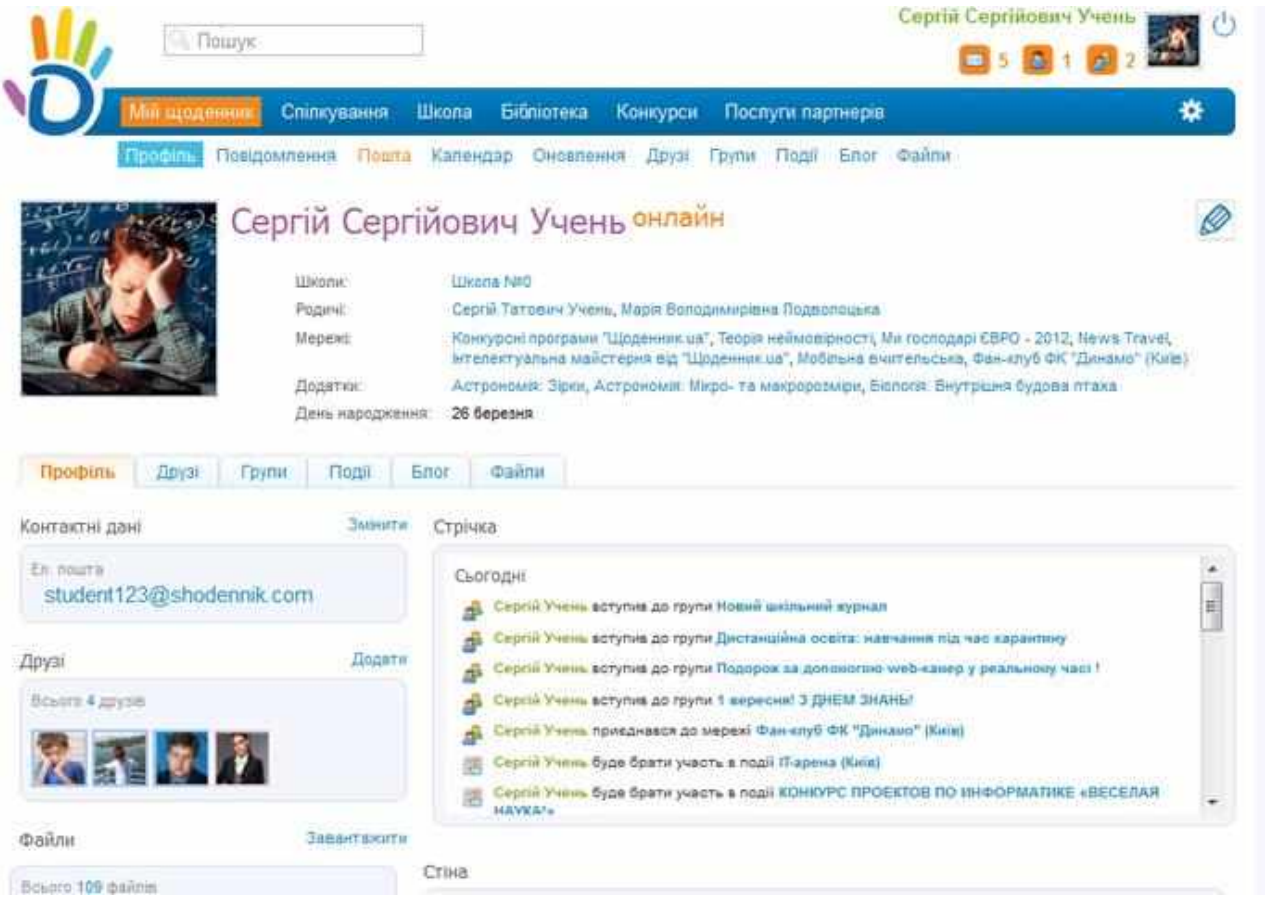

Рисунок 1.2 – Платформа shodennik.ua

Отже, дані продукти є досить цікавими, мають свої переваги та недоліки та дозволяють перенести частину навчального процесу в онлайн. Продукт, який розробляється в даному проєкті, відрізняється від представлених вище, а саме він має бути практичним, простим та не брати на себе зайві функції.

1.3. Постановка задачі

Задачею даного бакалаврського дипломного проєкту є аналіз актуальності проблеми створення системи, яка б забезпечувала можливість

ведення електронного щоденника, а також розробка програмного забезпечення для реалізації подібної системи.

Дану систему було вирішено розробляти у формі веб-додатка, адже такий вид програмного забезпечення є платформонезалежним та є доступним без встановлення додаткових компонентів.

1.4. Висновки

В даному розділі було проаналізовано чи існує необхідність створення певної системи для ведення електронного журналу. Дану систему було вирішено створювати саме у вигляді веб-орієнтовано додатку, що пояснюється зручністю такого виду додатків для користувачів.

Також було проаналізовано вже створені додатки, які мають схожий функціонал. Розроблений веб-додаток буде відрізнятися від уже існуючих аналогів, однією з його головних рис має бути простота та зручність для користувачів, а саме для вчителів, учнів та їх батьків.

РОЗДІЛ 2. АНАЛІЗ ТЕХНОЛОГІЙ, ЯКІ ВИКОРИСТОВУЮТЬСЯ ПРИ РОЗРОБЦІ ВЕБ-ДОДАТКІВ

2.1. Архітектурні особливості веб-додатків.

Архітектура веб-додатку це структура, яка описує взаємозв'язки та взаємодію між компонентами додатку такими як компоненти проміжного програмного забезпечення, базами даних та користувацьким інтерфейсом. Сучасна архітектура веб-додатків постійно розвивається та покращується як і в можливостях інтерфейсу (front-end) так і в можливостях серверної частини (back-end). Завдяки великому різноманіттю задач, які реалізуються веб-додатками існує досить велика кількість підходів до розробки такої архітектури додатку, яку була б доцільна саме при вирішенні конкретної задачі.

Від вибору конкретної архітектури залежить надійність, масштабованість та продуктивність продукту, що розробляється. Розглянемо деякі шаблони архітектури.

Архітектура клієнт-сервер (Client-server). В даній архітектурі існує два типи пристроїв: клієнт та сервер. Клієнтами називаються віддалені комп'ютери, які надсилають запити до серверу та отримують від нього відповіді. Сервер очікує запит від клієнту, обробляє його та відправляє відповідь клієнту. [3] Клієнтом може виступати будь-який персональний комп'ютер, сервером зазвичай є комп'ютер з досить потужними обчислювальними можливостями, яких вистачить для опрацювання запитів від багатьох користувачів.

Зазвичай в даному типі архітектури виділяють два рівні: рівень представлення (клієнт) та прикладний рівень (сервер). Іноді також виділяють і третій рівень – рівень зберігання даних (рис 2.1), до якого відноситься сервер бази даних.

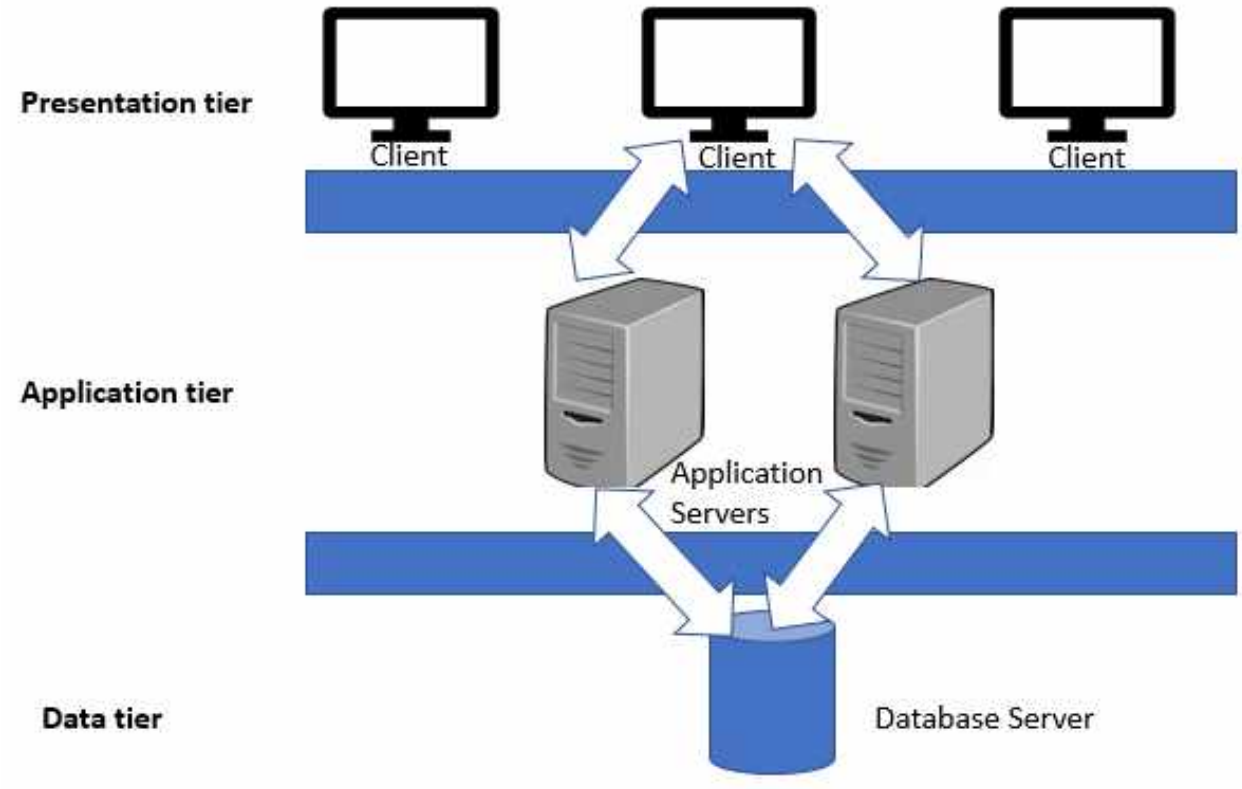

Рисунок 2.1 – Трирівнева модель архітектури клієнт-сервер.

Дана архітектура має досить багато переваг:

- Висока ступінь захисту, яка пояснюється тим, що всі операції з даними проводяться на сервері;
- Централізованість – всі ресурси контролюються окремим виділеним сервером;
- Масштабованість – кількість клієнтів та серверів може бути змінена в будь-який момент;
- Простота резервного копіювання, яка пояснюється тим, що всі дані зберігаються в одному місці.

Головним недоліком є перенавантаження серверів, яке може виникнути при занадто великій кількості запитів від клієнтів.

Комунікація між сервером відбувається за допомогою повідомлень типу запит-відповідь. Сучасна комунікація веб-додатків відбувається на основі REST

архітектури, котра базується на протоколі HHTP і зазвичай повертає дані у форматі JSON.

REST архітектура базується на таких принципах:

- Єдиний інтерфейс – завдяки використанню спрощується архітектура додатку в цілому і також покращується сприйняття взаємодії компонентів;
- Кешована архітектура – необхідно, щоб дані отримані у відповідь на запит були помічені як такі, що кешуються або не кешуються. Кешування даних необхідне для зменшення кількості запитів і зменшення навантаження на сервер, але потрібно звернути увагу, що постійне кешування впливає на актуальність відображених даних;
- Незалежність від стану – сервер не має запам’ятовувати стан користувача між запитами, це означає що відповідь від сервера не залежить від попередніх запитів. Стан може зберігатися лише на клієнті;
- Багаторівнева архітектура – клієнту невідомо чи є сервер до якого він звертається кінцевим;
- Зручне подання даних – зазвичай дані об’єкта передаються у форматі JSON або XML.

Найчастіше використовуються такі типи HTTP-запитів REST-архітектури: Get – для отримання даних, Post – для створення, Put – для зміни, Delete – для видалення.

Також в останні роки набирає популярність технологія WebSocket. WebSocket – це мережевий протокол, який забезпечує канал зв’язку по TCP-з’єднанню, призначений для встановлення зв’язку в режимі реального часу між сервером та клієнтом.

Даний протокол було розроблено для застосування у веб-додатках, але його можна також використати в будь-якому додатку, побудованого на принципах архітектури клієнт-сервер.

Необхідність розробки даного протоколу через потребу оновлення відображених даних у режимі real-time. До цього вся взаємодія клієнта відбувалася за допомогою HTTP запитів та відповідей на них. Такі запити не могли повністю задовольняти вимогам по швидкодії.

Розглянемо процес встановлення з'єднання для протоколу WebSocket (рис2.2). Для встановлення каналу клієнтом та сервером використовується протокол схожий на HTTP. На стороні клієнта формується особливий HTTP-запит, на який відповідає сервер. Після відповіді від сервера встановлюється двостороннє з'єднання[4].

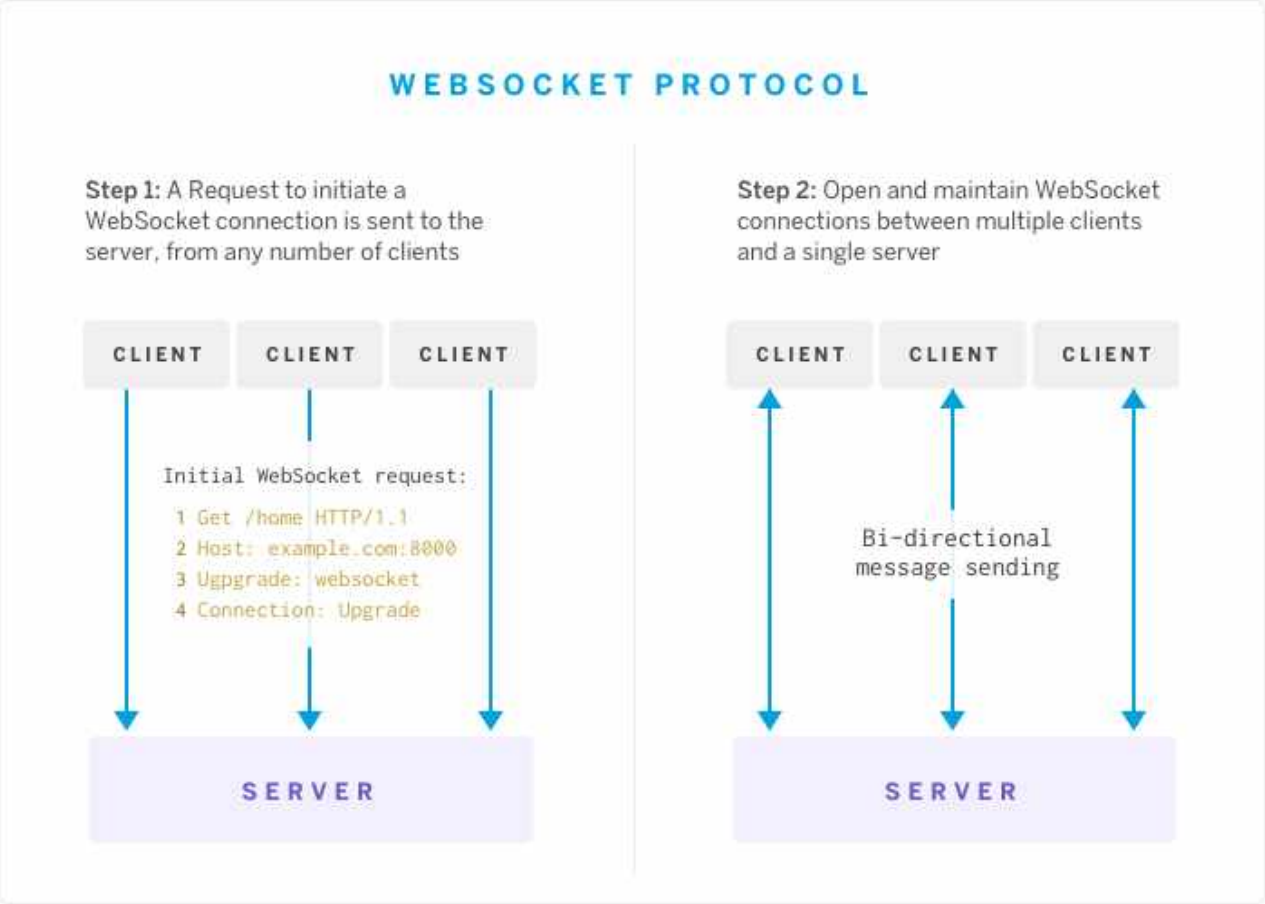

Рисунок 2.2 – Модель роботи WebSocket.

Дану технологію доцільно застосовувати в програмах, для роботи яких необхідне постійне з'єднання, наприклад, комп'ютерні ігри, додатки-чати тощо.

Клієнтська частина для даної технології реалізована в мові JavaScript, проте через її сучасність вона підтримується не усіма веб-браузерами. Клієнтська частина також повинна мати особливу реалізацію в мові програмування. Прикладом мов програмування, в яких вже реалізовано протокол WebSocket, є Node.js, Java, C#, C++ та деякі інші.

2.2. Порівняльний аналіз шаблонів проектування веб-додатків

Нині одним з найпопулярніших шаблонів проектування веб-додатків є MVC (Model-View-controller). Суть даного шаблону полягає в розбитті всього додатку на три головні логічні компоненти: модель (Model), представлення (View), контролер (Controller). Таке розбиття дозволяє відокремити інтерфейс користувача від прямої взаємодії з даними за допомогою третього елементу – контролера, в якому описана вся логіка роботи з даними.

Розглянемо загальну модель взаємодії компонентів MVC (рис 2.3). Користувач робить певний запит до контролера, який отримуючи запит звертається до моделі та займається опрацюванням даних, на основі такої обробки формується представлення, яке користувач отримує як відповідь на свій запит[5].

Розглянемо детально всі компоненти MVC.

Модель

Модель зберігає інформацію про властивості даних і напряму працює з ними, зазвичай використовуючи для цього доступ до певної бази даних. В кожній моделі має бути присутня інформація лише про одну певну частину даних. Для деяких проектів цього достатньо, але в більшості є проектів в моделі

також присутня логіка опрацювання тієї частини даних, за яку відповідає саме ця конкретна модель.

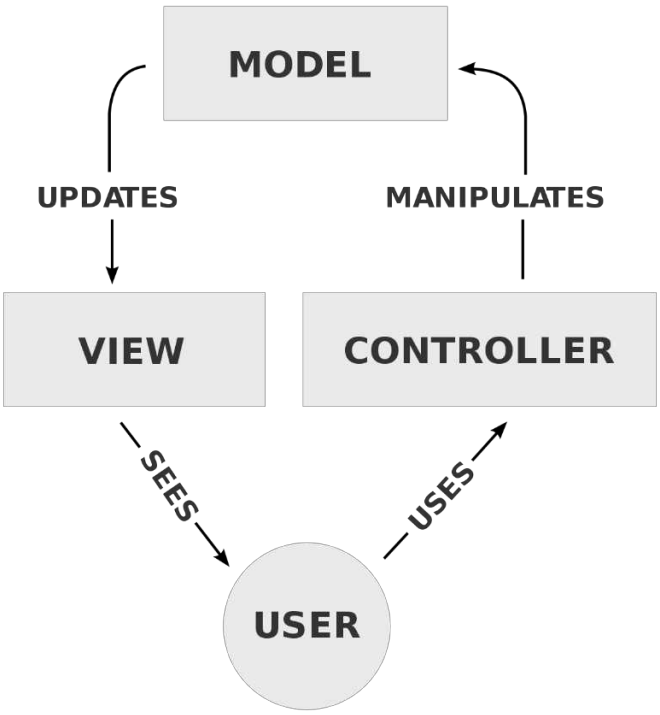

Рисунок 2.3 – Модель взаємодії компонентів шаблону MVC.

Одним з найбільш поширених прикладів можна назвати модель User (користувач). В такому прикладі модель буде зберігати інформацію про те, які властивості має User, наприклад, ім’я, роль і т.д. Крім цього в моделі також буде міститися певна логіка, яка пов’язана саме з цією моделлю.

Модель повинна бути повністю незалежною від інших частин програми. Модель абсолютно нічого не знає про контролер та представлення.

Представлення

Представлення відноситься до частини інтерфейсу програми та відповідає за візуалізацію інформації. В ньому зберігається розмітка, а також логіка відображення. Представлення формуються на базі даних, отриманих з моделі. Представлення формує запит від користувача на отримання інформації від моделі, після чого показує користувачу отриману відповідь. Представлення

не може напряму впливати на модель, тобто воно має доступ лише читання даних.

Для побудови представлень використовуються такі технології як HTML, CSS, JavaScript, а також деякі фреймворки.

Представлення також має бути відокремлене від інших компонентів: контролера та моделі. В представлення недопустимо додавати логіку маніпулювання з даними.

Контролер

Контролер є третьою частиною даного шаблону проектування і слугує для сполучення моделі та представлення. Також контролер фактично організовує зв'язок між користувачем та системою та використовує модель та представлення для реалізації необхідної відповіді на дії користувача. Саме на рівні контролера реалізується фільтрація отриманих вхідних даних та авторизація користувача – перевірка прав доступу на виконання тих чи інших дії з даними.

Тобто контролер має проводити підготовку даних для представлення, а також логіку оновлення даних на основі взаємодії з користувачем. Контролер не має довго зберігати дані, а також не повинен напряму оперувати DOM – моделлю.

Шаблон MVC реалізований на багатьох мовах програмування у вигляді окремих бібліотек або фреймворків, проте навіть не маючи таких бібліотек можливо реалізувати даний шаблон, для цього лише необхідно розбити текст програми на відокремлені за логікою модулі, повторюючи логіку шаблону MVC[6].

Розглянемо переваги та недоліки використання шаблону MVC. До переваг можна віднести наступні особливості:

- Відносно простий процес налагодження роботи програми – завдяки тому, що структурно програма розділена на окремі логічні елементи, можливо тестувати кожен елемент окремо;

- Можливість ведення паралельної розробки кількома програмістами, що скорочує загальний час створення програми;
- Слабкий зв’язок між окремими елементами (моделями, представленнями, контролерами) – спрощує процес додавання або видалення певних частин програми.

До недоліків можна віднести:

- Достатньо велика кількість технологій, які потрібно знати для створення програми – зазвичай контролер, представлення та моделі створені за допомогою різних технологій;
- Складність розуміння самої архітектури – введення кількох рівнів абстракції ускладнює розуміння роботи програми.

MVC виявися досить популярним, тому на його базі було створено декілька різновидів шаблонів проєктування. Розглянемо найпопулярніші з них.

Model-View-Presenter (MVP) – в даному шаблоні контролер було замінено на представник (Presenter). Елемент представник є посередником (рис2.4), так само як контролер в MVC[7].

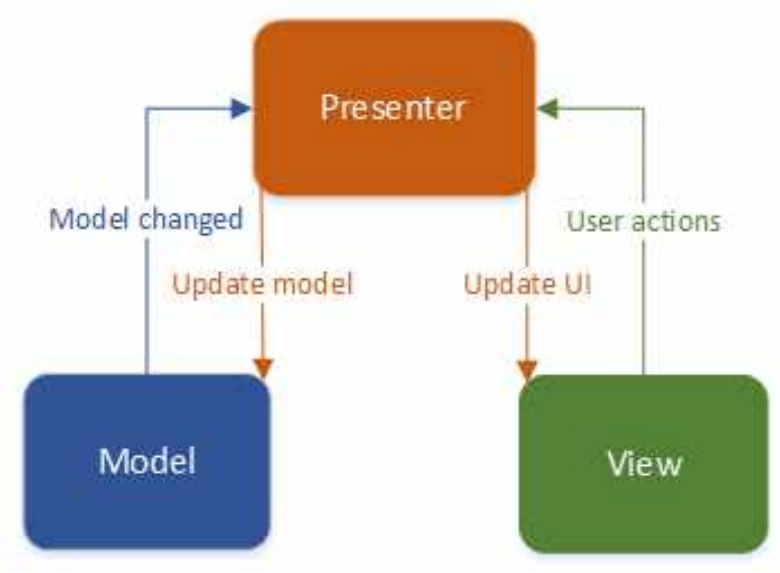

Рисунок 2.4 – Модель взаємодії компонентів шаблону MVP.

До обов’язків представника відноситься керування подіями (наприклад, натиск кнопки на клавіатурі) користувацького інтерфейсу. Крім того, представник відповідає за синхронізацію моделі та представлення, а також саме в ньому зберігається вся бізнес-логіка програми.

Model-View-ViewModel (MVVM) (рис2.5) – був розроблений як модифікація MVP.

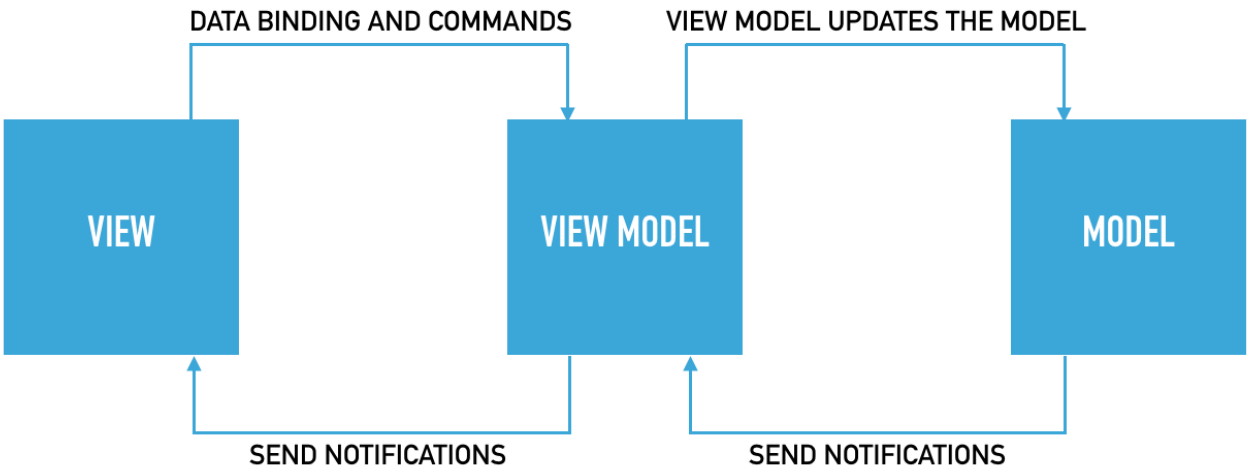

Рисунок 2.5 – Модель взаємодії компонентів шаблону MVVM.

MVVM зазвичай використовується у випадках, коли під час розробки присутнє ‘зв’язування даних’ (data binding). Така присутність вимагає двостороннього зв’язку між моделлю ті представленням, що не відповідає вимогам класичного MVC. Модель-представлення (ViewModel) є одночасно абстракцією представлення та обгорткою даних з моделі, які мають бути ‘зв’язані’[8]. Це означає, що в ній зберігається модель приведена до представлення. Крім того, модель-представлення зберігає логіку, яку представлення використовує для впливу на модель.

2.3. Вибір типу бази даних та системи керування базами даних (СКБД)

Дуже важливою частиною майже кожного веб-додатку є база даних. В базі даних зберігається вся змінна інформація, необхідна для функціонування веб-додатку. Зазвичай БД розташовується на окремому сервері. Такі сервери мають бути достатньо потужними, тому зазвичай являють собою багатопроцесорними системами. В серверах баз даних використовуються масиви дисків типу RAID. Використання таких дисків забезпечує досить високу надійність збереження даних, наприклад, можливе відновлення даних навіть при виході з ладу одного або кількох дисків.

Необхідно обрати СУБД, яка б могла дозволити редагування записів кількома користувачами одночасно, крім того необхідна підтримка збережених процедур та індексації, яка необхідна для збільшення швидкодії виконання запиту.

В загальному бази даних поділяються на реляційні та нереляційні (NoSql). Реляційні БД є історично більш популярні, але останнім часом часта NoSql БД збільшилась за даними ресурсу http://db-engnes/en/ranking-trend (рис 2.6).

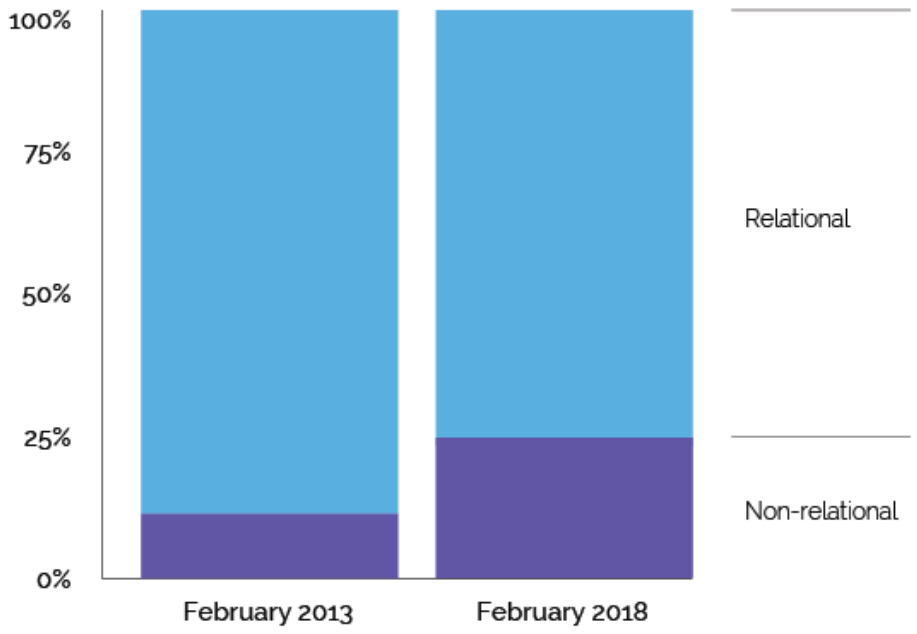

Рисунок 2.6 – Популярність реляційних та нереляційних БД.

Реляційні БД було розроблено в 70-х роках 20ст. для зберігання даних відповідно до схеми, яка б дозволяла відображувати дані у вигляді таблиць з рядками та стовпчиками. Реляційна БД являє собою набір таблиць, кожна з яких має свою схему, в якій зберігаються фіксовані поля та їх типи даних, які будуть мати дані в таблиці. Всі СКБД надають можливості вибору, додавання, видалення та редагування даних.

Таблиці в реляційних БД мають ключі, які зв'язані з ними та використовуються для ідентифікації конкретних стовпчиків або рядків таблиці, а також наявність ключів дозволяє зменшити час доступу до конкретного рядка або стовпчика[9].

Цілісність даних особливо важлива для реляційних БД, тому системи керування БД використовують велику кількість обмежень для забезпечення надійності і точності даних, які зберігаються в таблицях.

Існує велика кількість різноманітних реляційних БД. Найпопулярніші з них (рис 2.7):

- Oracle – БД Oracle (зазвичай її називають Oracle RDBMS або просто як Oracle) - це багатомодельна система управління базами даних, що виробляється та продається корпорацією Oracle;
- MySQL - це СКБД з відкритим кодом на основі структуризованої мови запитів (SQL). MySQL працює практично на всіх платформах, включаючи Linux, UNIX та Windows;
- Microsoft SQL Server - це СКБД, яка підтримує широкий спектр програм для обробки транзакцій, бізнес-аналітики та аналітики в корпоративних IT-середовищах;
- PostgreSQL - , часто просто Postgres - це об'єктно-реляційна системою управління базами даних (ОРСКБД) з акцентом на розширюваність та відповідність стандартам;
- DB2 - СКБД, призначена для ефективного зберігання, аналізу та отримання даних.

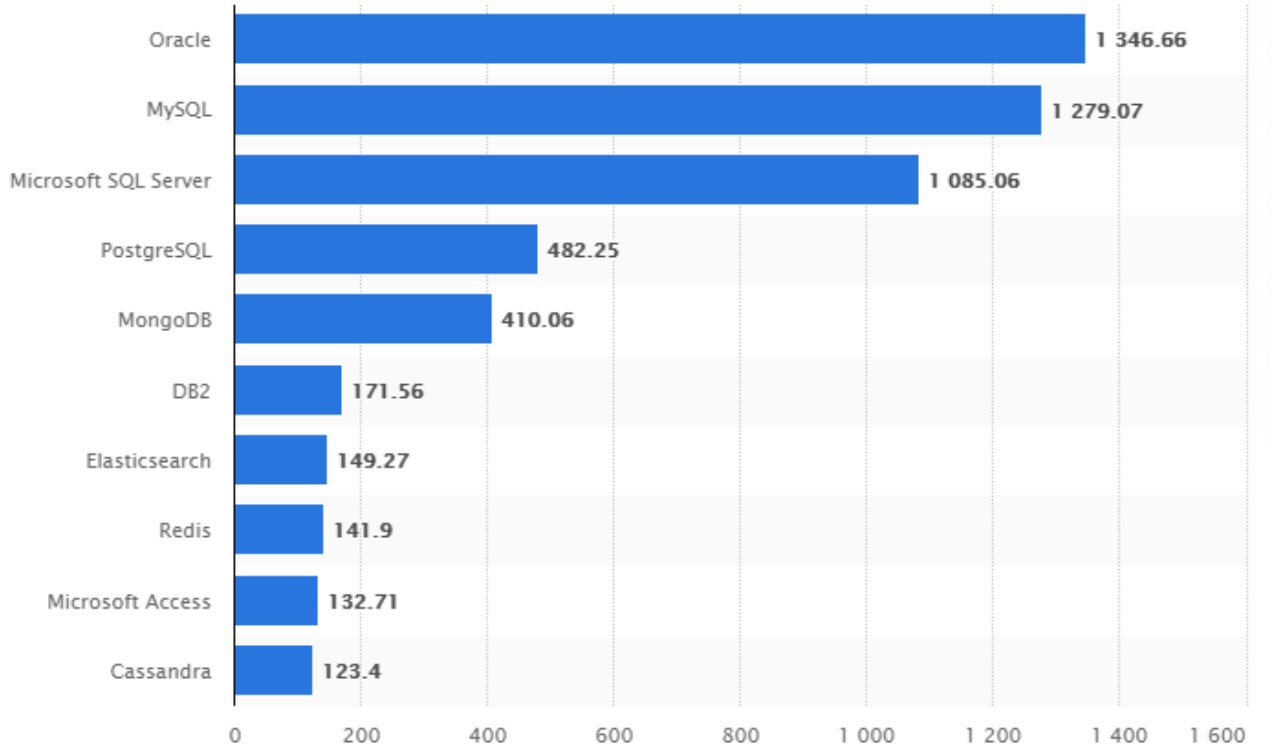

Рисунок 2.7 – Популярність реляційних БД за даними веб-ресурсу db-engines.com.

Переваги реляційних БД.

- Реляційні бази даних є добре задокументованими та зрілими технологіями, а реляційні СКБД продаються та підтримуються низкою створених корпорацій;
- Стандарти SQL є чітко визначеними та загальноприйнятими;
- Великий пул кваліфікованих розробників має досвід роботи з SQL та реляційними СКБД;
- Всі реляційними СКБД сумісні з обмеженнями ACID, тобто вони задовольняють вимогам атомності, консистенції, ізоляції та довговічності.

Недоліки:

- Реляційні СКБД можуть працювати лише частково або взагалі не можуть з неструктурованими або слабко-структурованими даними

через обмеження, які накладаються схемою та типом. Це робить їх непридатними для великих аналітичних навантажень;

- При переміщенні однієї реляційної БД на іншу, схеми та типи, як правило, повинні бути однаковими між вихідними та цільовими таблицями для міграції (обмеження схеми).
- З багатьох причин складні набори даних або ті, що містять записи змінної довжини, як правило, важко обробляти схемою реляційної бази даних.

Найбільш популярними реляційними СКБД є

Нереляційні бази даних (NoSql) стали популярною альтернативою реляційним базам даних, оскільки веб-додатки ставали все складнішими. Бази даних NoSql приймати найрізноманітніші форми. Однак критична відмінність між NoSql і реляційними базами даних полягає в тому, що схеми реляційних СКБД жорстко визначають, як всі дані, вставлені в базу даних, повинні бути введені і складені, тоді як бази даних NoSql можуть бути схематичними, дозволяючи зберігати і маніпулювати неструктурованими і слабоструктурованими даними.

В нереляційних БД деякі сутності можуть належати до декількох категорій. Наприклад, Couchbase - це і база даних документів, і база даних «ключ—значення».

Існує кілька типів нереляційних (NoSql) БД, кожен з яких має кілька реалізацій(рис. 2.8)

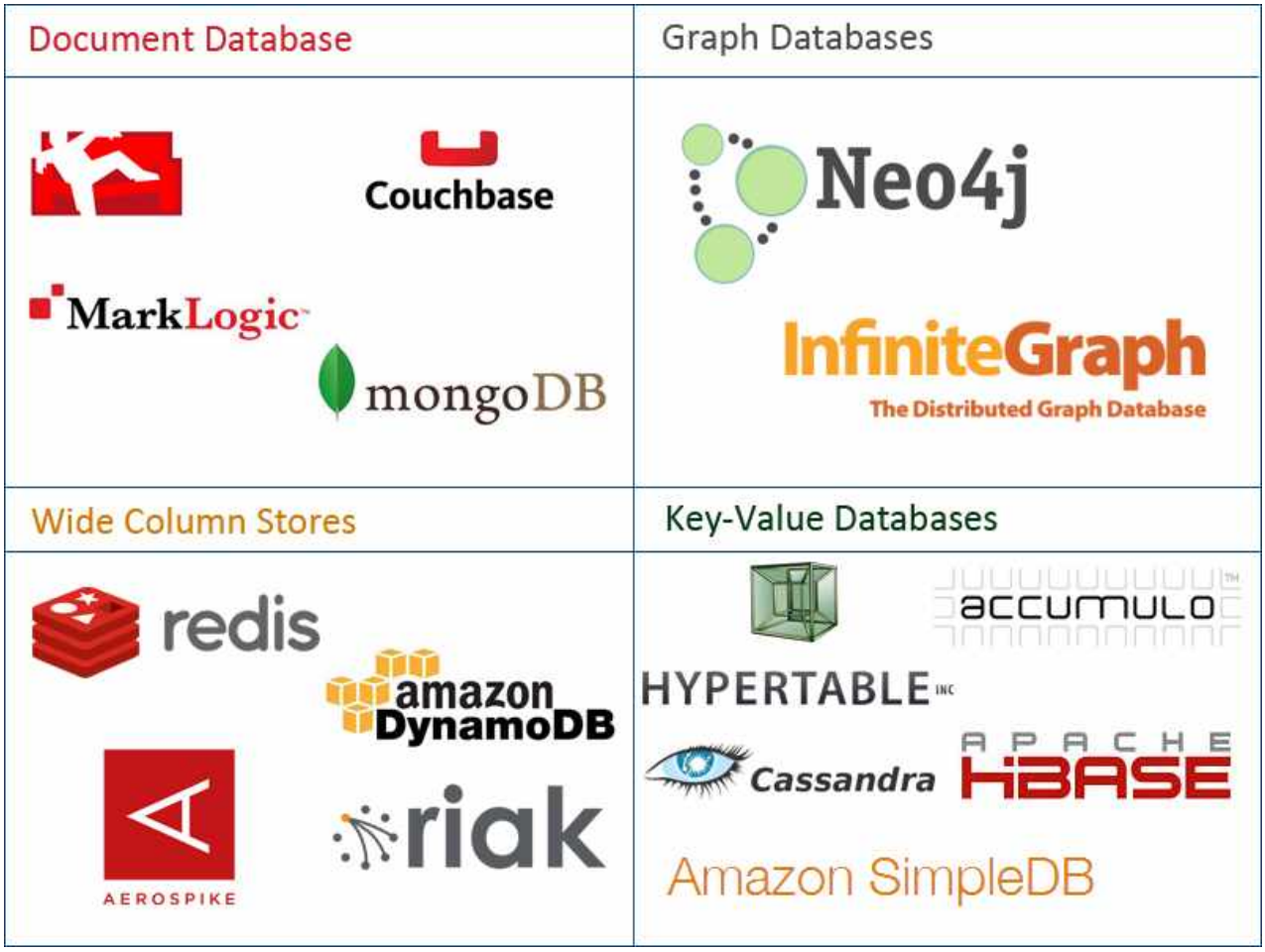

Рисунок 2.8 – Приклад нереляційних (NoSql) баз даних.

Бази даних «ключ—значення» такі як Redis та Amazon DynamoDB, надзвичайно прості СКБД, які зберігають лише пари ключових значень і забезпечують основну функціональність для отримання значень, пов'язаних з відомим ключем.

Простота база даних типу «ключ—значення» робить ці системи управління базами даних особливо пристосованими до вбудованих баз даних, де збережені дані не є особливо складними, а швидкість має першорядне значення.

Сховище з широким стовпчиком, такі як Cassandra, Scylla та HBase - це схемо-агностичні системи, які дозволяють користувачам зберігати дані в сімействах стовпців або таблицях, один рядок яких можна розглядати як запис - багатовимірне значення ключа сховища.

Ці рішення розроблені з метою масштабування для управління петабайтами даних на багатьох тисячах товарних серверів у масивній розподіленій системі.

Сховища з широким стовпчиком, як Scylla та Cassandra, технічно використовують варіант SQL під назвою CQL для визначення даних та маніпулювання, що робить їх простими для тих, хто вже знайомий з реляційними СКБД.

Документо-орієнтовані системи керування базами даних, включаючи MongoDB та Couchbase, - це системи без схем, які зберігають дані у вигляді документів JSON. Сховища документів схожі на бази даних типу «ключ—значення» або сховища з широким стовпчиком, але назва документа - це ключ, а вміст документа - значення.

У документо-орієнтованих СКБД окремі записи не потребують рівномірної структури, можуть містити багато різних типів значень і можуть бути вкладеними. Ця гнучкість робить їх особливо придатними для управління напівструктурованими даними в розподілених системах.

Графові БД, такі як Neo4J та Datastax Enterprise Graph , представляють дані як мережу пов'язаних вузлів або об'єктів, щоб полегшити візуалізацію даних та аналітику графіків.

Вузол або об’єкт у графовій БД містять дані у вільній формі, які пов'язані між собою зв'язками та згруповані відповідно до міток. Програмне забезпечення для управління базами даних (СУБД), орієнтованих на графи, розроблене з акцентом на ілюструванні зв’язків між точками даних.

Як результат, бази графові БД зазвичай використовуються, коли аналіз взаємозв'язків між неоднорідними точками даних є кінцевою метою системи, наприклад, у запобіганні шахрайствам, вдосконаленій діяльності підприємства або оригінальним графіком друзів Facebook.

Пошукові системи , такі як Elasticsearch, Splunk та Solr, зберігають дані за допомогою JSON-документів, що не містять схеми. Вони схожі на документо-

орієнтовані СКБД, але з більшим акцентом роблять доступ до неструктурованих або напівструктурованих даних легко доступними за допомогою текстового пошуку з рядками різної складності.

Оскільки існує так багато типів і різноманітних застосунків баз даних NoSQL, важко виокремити спільні переваги, але в загальному випадку до переваг можна віднести такі якості:

- Моделі даних без схем є більш гнучкими та легшими в управлінні;
- Бази даних NoSql, як правило, є більш горизонтально масштабованими та відмовостійкими;
- Дані можна легко розподілити по різних вузлах. Щоб поліпшити доступність та / або переносимість розділів, ви можете вибрати, щоб дані про деякі вузли були "з часом узгоджені ".

Недоліки також залежать від типу бази даних. В основному:

- Бази даних NoSql, як правило, менш широко прийняті та зрілі, ніж реляційні СКБД, тому часто потрібна спеціальна експертиза;
- Існує діапазон форматів і обмежень, характерних для кожного типу бази даних.

В даному бакалаврському проєкті всі дані будуть добре структурованими, тому необхідно вибрати таку СКБД, яка б відповідала принципам ACID. Ці принципи включають в себе:

- Атомарність – під даною властивістю мається, що транзакція може бути лише такою, яка відбулася або такою, що не відбулася взагалі, середнього стану, який би передбачав часткове виконання транзакції не існує;
- Узгодженість – гарантується відповідність усіх даних, які мають бути дійсними згідно з усіма накладеними обмеженнями;
- Ізольованість – дана властивість забезпечує одночасне виконання кількох транзакцій, не призводячи до неузгодженості стану бази даних;

- Довговічність – дана властивість гарантує, що після завершення транзакції оновлення та зміни БД зберігаються та записуються на диск.

Реляційні СКБД повною мірою відповідають принципам ACID, тому для виконання даного дипломного проєкту було обрано реляційну СКБД, а саме Microsoft SQL Server.

2.4. Висновки

В даному розділі було проаналізовано загальні принципи, які використовуються при розробці веб-додатків. Було проаналізовано принцип роботи архітектури типу клієнт-сервер. Більшість веб-додатків проєктуються саме з використанням даного типу архітектури, веб-додаток, який розробляється в даному бакалаврському проєкті також використовує клієнт-серверну архітектуру.

Було розглянуто кілька типів шаблонів проєктування, які використовуються при розробці веб-орієнтованих додатків серед яких MVC, MVP, MVVM. Було описано принцип роботи кожного з них, а також описано їх переваги та недоліки. Проаналізувавши дану інформацію, як шаблон для веб-додатку, що розробляється, було обрано MVC (Model-View-Controller). Даний паттерн проектування було обрано через те, що даному в шаблоні існує поділ всього проєкту на окремі частини, завдяки чому даний шаблон є досить простим для розуміння, а також не потребує великих часових затрат для вивчення. Крім того, проєкти, які розроблені з використанням даного шаблону, досить просто доповнювати новими компонентами, що надає досить великі можливості для майбутнього покращення існуючого функціоналу, а також впровадження нового.

Так як для повноцінною роботи веб-додатку для ведення журналу успішності школярів необхідна наявність бази даних було описано два основних, з нині існуючих, типів баз даних: реаляційні(SQL) та нереляційні(NoSQL), було проаналізовано їх відмінності, а також переваги та недоліки кожного типу.

З огляду на те, що дані, які будуть зберігатися та використовуватися, є добре структурованими, було прийнято рішення обрати таку СКБД, яка б відповідала принципам ACID. Даним принципам відповідають реляційні СКБД, тому було обрано СКБД цього типу, а саме Microsoft SQL Server.

РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ ДЛЯ ВЕДЕННЯ ЖУРНАЛУ УСПІШНОСТІ ШКОЛЯРІВ

3.1. Вибір моделі життєвого циклу програмного забезпечення

В загальному розумінні життєвий цикл програмного забезпечення – це період часу, який починається з моменту прийняття рішення про необхідність створення програмного додатку до моменту його повного виведення з використання. Нині існує кілька моделей і вибір цієї модель є досить важливою частиною процесу розробки програмного продукту, адже дотримуючись певної моделі простіше поетапно реалізовувати проект.

Найпопулярніші моделі на сьогоднішній день:

- Спіральна модель;
- Ітераційна модель;
- Каскадна (водоспадна) модель;
- Інкрементна модель.

Найпростішою для розуміння та у використанні є каскадна модель, крім того це перша з представлених моделей. У даній моделі весь процес розробки програмного забезпечення розділений на окремі фази. Результат однієї фази є вхідними даними для наступної фаз. Це означає, що будь-яка фаза процесу розвитку починається лише за умови завершення попередньої фази. Модель водоспаду - це послідовний процес проектування, в якому прогрес сприймається як постійно протікає вниз (як водоспад) через такі фази: визначення вимог, проектування архітектури системи, реалізація, інтеграція та тестування, впровадження системи, підтримка та супровід (рис 3.1).

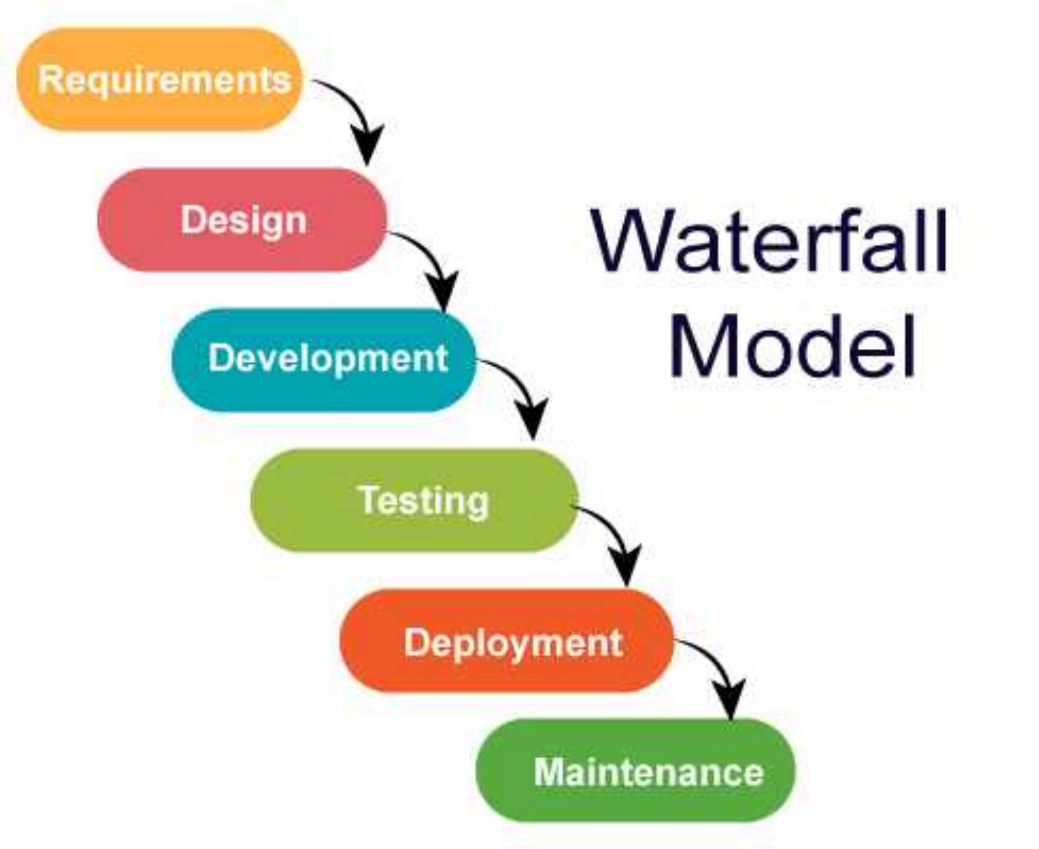

Рисунок 3.1 – Каскадна модель

Послідовні фази в моделі водоспаду

- Визначення вимог: Перший етап передбачає розуміння того, що потрібно розробити та яка його функція, призначення тощо. На даному етапі специфікуються дані, які мають надійти на вхід та, які мають бути отримані виході з фінального продукту.
- Проектування архітектури системи: на цій фазі вивчаються технічні вимоги з першого етапу і готується проектування системи. Дизайн системи допомагає у визначенні апаратних та системних вимог, а також допомагає визначити загальну архітектуру системи.
- Реалізація: використовуючи результати отримані на попередньому кроці, розробляються окремі блоки програми, котрі можна назвати юнітами. Перевіряється функціональність кожного блоку, такий процес називається юніт-тестуванням.

- Інтеграція та тестування: Усі блоки, які були розроблені на етапі реалізації, інтегруються в систему після тестування кожного блоку. Розроблене програмне забезпечення повинно пройти повне тестування програмного забезпечення, щоб з'ясувати, чи є якісь недоліки чи помилки. Тестування проводиться так, щоб клієнт не стикався з жодними проблемами під час встановлення програмного забезпечення.
- Впровадження системи: Після того, як буде проведено функціональне та нефункціональне тестування, розроблене програмне забезпечення буде розгорнуто в середовищі споживачів або випущено на ринок.
- Підтримка та супровід: Цей етап відбувається після встановлення та включає в себе зміни в системі або окремому компоненті для зміни атрибутів або підвищення продуктивності. Ці зміни виникають або через запити на зміни, ініційовані замовником, або дефекти, виявлені під час використання живої системи. Клієнту надається регулярне обслуговування та підтримка розробленого програмного забезпечення.

Усі ці фази є каскадними одна до одної, коли прогрес сприймається як постійно протікає вниз (як водоспад) крізь фази. Наступний етап починається лише після того, як визначений набір цілей буде досягнуто для попереднього етапу, тому дану модель називають каскадною або ж водоспадною.

Переваги каскадної моделі:

- Можливість здійснювати розбиття розробки на окремі частини з контролем кожної частини. Це надає можливість встановлення чіткого графіку розробки з визначеними дедлайнами на кожний етап.
- Каскадна модель розбита на прості та зрозумілі етапи, що робить її легкою для розуміння.

- Легкість в управлінні за рахунок жорсткості моделі - кожна фаза визначає результат, який має бути отриманий на виході, а також процес перевірки даного результату.
- У цій моделі фази обробляються та завершуються одна за одною і вони не перетинаються. Каскадна модель добре працює для невеликих проектів, де вимоги добре визначені.

Недоліки моделі водоспаду:

- Важко оцінити час і витрати на кожну фазу процесу розробки.
- Після того, як заявка знаходиться на етапі тестування, дуже важко повернутися назад і змінити щось, що не було продумано на етапі концепції.
- Не дуже вдала модель для складних та об'єктно-орієнтованих проектів.
- Не підходить для проектів, де вимоги мають помірний та високий ризик зміни.

Так як використання даної модель є простим для розуміння та досить зручним при розробці досить невеликих проектів, особливо тих над, з якими працює лише один розробник, при розробці даного веб-додатку було прийнято рішення використовувати саме каскадну модель життєвого циклу програмного забезпечення, також було детально розглянуто кожен етап даної моделі.

3.2. Вибір інструментів для програмної реалізації веб-додатку

Для розробки веб-додатку використовувалися різні інструменти. Це зумовлено тим, що проект було поділено на кілька частин, кожна з яких потребувала використання різного стеку технологій.

Для реалізації більшої частини проекту, а саме частини з реалізацією логіки була використана мова програмування C# та фреймворк ASP.NET MVC

Framework. Вибір цього фреймворку було спричинено наступними його перевагами:

- Фреймворк має повний контроль над згенерованою HTML розміткою сторінки;
- Має просту інтеграцію з мовою програмування JavaScript;
- Має уже вбудовану схему MVC, завдяки якій розділяється логіка, інтерфейс та доступ до БД;
- Має чітко розділений контролер та представлення. Такий розподіл забезпечує виконання принципу SoC (англ. separation of concerns) – розподіл відповідальності;
- Забезпечує досить високий рівень безпеки завдяки наявності вбудованих засобів автентифікації;
- Має вбудовані функції кешування;
- .Net Framework швидко надсилає попередження про витоки пам’яті, нескінченні цикли та іншу неправильну поведінку, негайно завершує їх роботу та перезавантажує заново;
- Висока продуктивність. ASP.NET має перевагу перед іншими технологіями, заснованими на сценаріях, завдяки автоматичному компілюванню коду на стороні сервера до файлів DLL на веб-сервері. ASP.NET підвищує свою ефективність, використовуючи переваги компіляції JIT, профілювання часу виконання, автоматичне управління пам’яттю та оптимізацію ресурсів, обробку виняткових ситуацій, раннє прив'язування та краще кешування;
- Велика кількість підтримуваних мов. ASP.NET підтримує всі мови, сумісні з CLR, до них відноситься більше 25 мов і має вбудовану підтримку VB.NET, C#, F#. Це надає програмістам безпрецедентний вибір мови та дозволяє зосередитись лише на процесі розробки. Не потрібно, щоб розробник витрачав час на

вивчення нових мов програмування, щоб завершити проект. В результаті економиться час і зусилля. Завдяки ASP.NET MVC можливо розробляти веб-програми на десятках найпопулярніших мов, зберігаючи відмінні показники.

- Спрощений процес публікації додатків. Розгортання веб-додатків стало набагато простіше за допомогою ASP.NET MVC. Весь додаток можна швидко розгорнути, просто скопіювавши його на сервер. Параметри конфігурації зберігаються у XML-файлі в програмі. Перед розгортанням кожен компонент може бути протестований незалежно від інших. Спрощене розгортання означає економію багато часу, яке можна використовувати для життєдіяльності процесу розробки програмного забезпечення. Також для публікації додатку непотрібно перезавантажувати веб-сервер.

Для розробки рівня представлення було використано синтаксис розмітки шаблонів Razor – для розмітки веб-сторінки, CSS3 - для налаштування стилів веб-сторінки, JavaScript – мова програмування на стороні клієнта.

Razor - синтаксис розмітки веб-шаблонів. Ідея Razor полягає в тому, щоб забезпечити оптимізований синтаксис для генерації HTML, використовуючи шаблонний підхід, орієнтований на код, з мінімальним переходом між HTML та кодом. Даний синтаксис використовується для створення динамічних веб-сторінок за допомогою мов програмування C# та VB.Net[10]. Дана розмітка виглядає як поєднання класичної HTML розмітки з елементами мови високого рівня C# (рис 3.2). Використовуючи Razor серверний код може створювати динамічний веб-контент на ходу, в той час як веб-сторінка записується в браузер. Коли викликається веб-сторінка, сервер виконує серверний код всередині сторінки, перш ніж повертати сторінку в браузер.

```
@{
    string[] members = { "Jani", "Hege", "Kai", "Jim" };
    int i = Array.IndexOf(members, value:"Kai") + 1;
    int len = members.Length;
    string x = members[2 - 1];
}
<html>
<body>
    <h3>Members</h3>
    @foreach (var person in members)
    {
        <p>@person</p>
    }
    <p>The number of names in Members are @len</p>
    <p>The person at position 2 is @x</p>
    <p>Kai is now in position @i</p>
</body>
</html>
```

Рисунок 3.2 – Приклад розмітки Razor

Дану технологію було обрано через наступні її переваги:

- Компактність, виразність і гнучкість: Razor знадає швидкий і гнучкий процес програмування. На відміну від більшості синтаксичних шаблонів, в даному випадку не потрібно переривати написання коду, щоб явно визначити серверні блоки коду в HTML. Парсер досить розумний, для того щоб зробити це за програміста, ґрунтуючись на отриманому тексті програми. В результаті цього синтаксис виходить компактним і зрозумілим, А також простим для написання;
- Легкий для освоєння: Razor простий в освоєнні і дозволяє за короткий час звикнути до нього;
- Razor не нова мова: Razor не ґрунтується на новій імперативній мові. Завдяки цьому розробники мають можливість використовувати вже існуючі навички роботи з C # / VB (або іншою

мовою). Завдяки цьому було отримано синтаксис розмітки шаблону, який дає вам приголомшливий процес побудови HTML, залишаючи вибір мови за програмістом;

- Підтримка Intellisense: Не дивлячись на те, що Razor розроблявся з урахуванням відсутності прив'язки до певного інструменту або редактору коду, він має підтримку у Visual Studio.

CSS (Cascading Style Sheet) – мова стилю сторінок, яка використовується для форматування макета веб-сторінки. CSS також можна використовувати для визначення стилів тексту, розмірів таблиць та інших аспектів веб-сторінок. За допомогою CSS веб-розробники створюють одноманітний вигляд на декількох сторінках веб-сайту. Замість визначення стилю кожної таблиці та кожного блоку тексту в HTML-коді сторінки, загальновживані стилі визначаються лише один раз у документі з CSS стилями. Після того, як стиль визначений у каскадному стилі, його може використовувати будь-яка сторінка, на яку посилається файл CSS. Крім того, CSS дозволяє легко змінювати стилі на декількох сторінках одночасно.

Крім того, було використано бібліотеку стилів Bootstrap. Дана бібліотека містить шаблони дизайну на основі CSS (іноді JavaScript) для кнопок, навігації, текстових елементів та інших елементів користувацького інтерфейсу. Дана бібліотека дуже спрощує та пришвидшує розробку інтерфейсної частини веб-додатку. Також за допомогою цієї бібліотеки можливо створити адаптивну веб-сторінку, розташування компонентів на якій залежало б від розміру екрана користувача.

JavaScript – мультипарадигмова мова програмування високого рівня, яка відповідає специфікації ECMAScript. Використовується для додання гнучкості та інтерактивності веб-сторінці.

Також було використану бібліотеку jQuery для мови JavaScript. Дана бібліотека дозволяє зручно керувати елементами DOM, створювати анімацію, а

також надсилати Ajax-запити. Саме функція створення Ajax-запитів застосовувалася для зв’язку клієнту з сервером та обміну між ними даними.

Саме з використанням цих технологій створюється структура рівня представлення.

Рівень представлення сама та частина веб-додатку, яку бачить користувач та з якою він взаємодіє. На даному рівні може знаходитись лише проста логіка, яка відповідає за відображення.

В шаблоні MVC представлення відокремлене від контролера та моделі, саме використовуючи ASP.NET MVC Framework, це дуже помітно, адже файли з реалізацією представлення є окремими та мають інше розширення файлу, а саме cshtml, яке відповідає формату Razor.

Також при розробці певного програмного продукту необхідно приділити увагу вибору IDE (англ. Integrated Drive Electronics) – середовищу розробки, оскільки вибір зручного IDE пришвидшить та спростить розробку. Для розробки даного веб-додатку було обрано інтегроване середовище розробки Visual Studio 2017 (рис 3.3).

Microsoft Visual Studio - це інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Дане середовище використовується для розробки комп’ютерних програм, а також веб-сайтів, веб-додатків, веб-служб та мобільних додатків. Visual Studio використовує платформи Microsoft для розробки програмного забезпечення, такі як API Windows, Windows Forms, Foundation Presentation Foundation, Windows Store та Microsoft Silverlight .

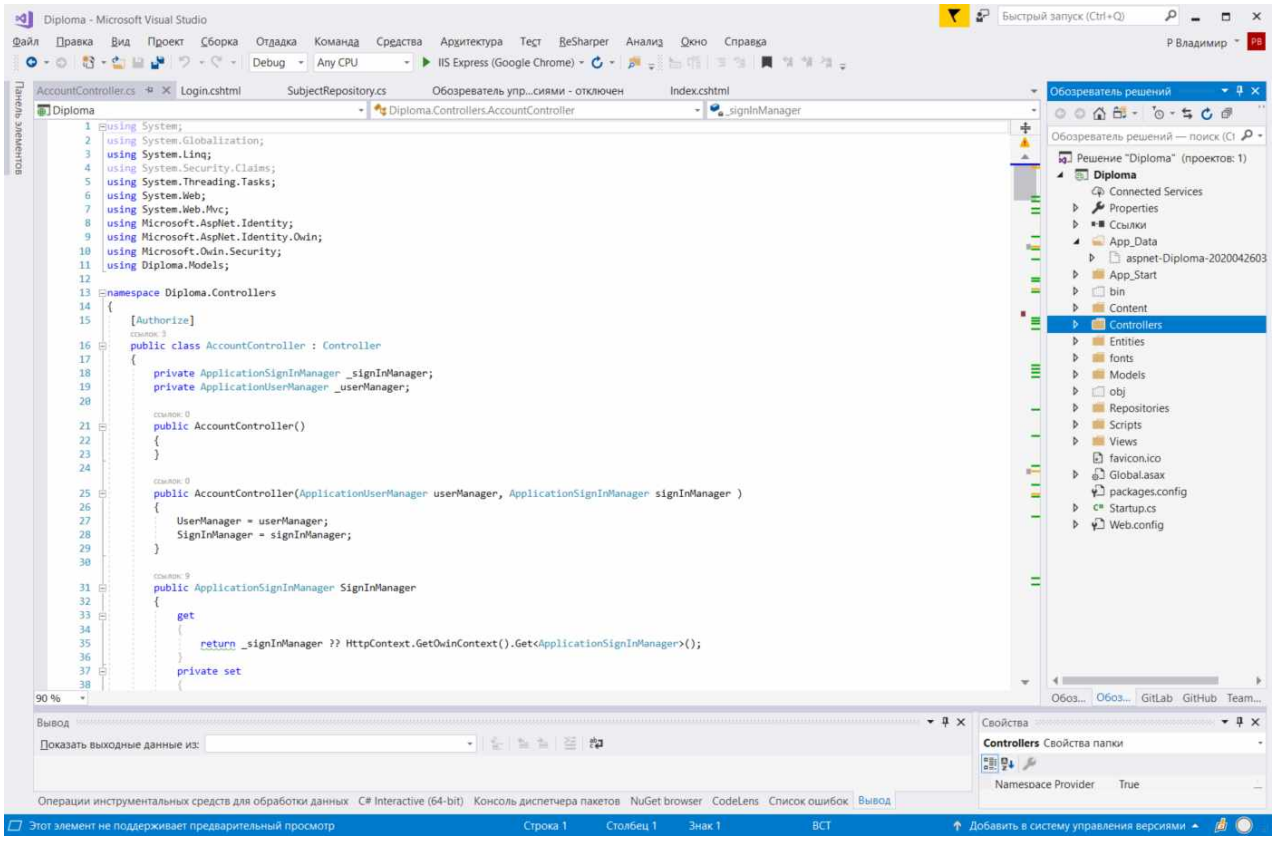

Рисунок 3.3 – Інтегроване середовище розробки Visual Studio 2017

Visual Studio включає редактор коду, що підтримує IntelliSense (компонент доповнення коду), а також рефакторинг коду. IntelliSense – допоміжний засіб для написання тексту програми, який перевіряє код в реальному часі, включає в себе такі можливості [11]:

- Список членів;
- Інформація про параметри;
- Короткі відомості;
- Закінчити слово.

Такі можливості допомагають отримувати додаткові відомості про код, що використовується, відслідковувати параметри при введенні, а також додавати виклики методів за допомогою натискання всього кількох клавіш.

Visual Studio також надає змогу відлагоджувати код, написаний мовами C#, C та багатьма іншими. До функцій відлагодження входить можливість встановлення точки зупинки (breakpoint) та встановлення різноманітних

налаштувань для цієї точки, відображення стеку викликів, перегляд значень змінних надає можливість їх замінити. Такий функціонал дуже допомагає при виправленні помилок.

Даний продукт забезпечує інтеграцію систем контролю версій таких як git та tfs. За допомогою такого функціоналу можливо фіксувати всі зміни, які відбувалися при написанні програми, створюється історія завдяки чому, в разі додання неуспішних змін, є можливість повернути стан програми до однієї з попередніх версій.

3.3. Опис архітектури веб-додатку

Оскільки архітектура даного веб-додатку розроблена на базі паттерну MVC, весь проект поділено на окремі модулі, кожен з яких відповідає за окрему веб-сторінку додатку. Кожен такий модуль складається з контролера, представлень та моделей, пов’язаних саме з конкретним контролером. Оскільки фреймфорк ASP.NET MVC надає можливість створювати часткові представлення до кожного контролера відноситься кілька таких представлень. Завдяки такому підходу зручно розділяти все представлення на кілька окремих структурних блоків, кожен з яких може бути відображеним або ні. Для прикладу, на веб-сторінці міститься блок з фільтрами, його зручно винести у часткове представлення та відображувати у разі необхідності.

Розглянемо модуль, який відповідає за авторизацію. Даний модуль складається з наступних компонентів:

- AccountController.cs - контролер;
- Login.cshtml – часткове представлення, яке бачить користувач при вході;
- Register.cshtml – часткове представлення, яке користувач бачить при реєстрації;

- ResetPassword.cshtml – часткове представлення для скидання паролю:
- UserModel.cs – модель користувач.

Даний модуль було розроблено на основі системи авторизації да аутентифікації ASP.NET Identity. Дана система має кілька видів аутентифікації, а саме:

- Аутентифікація індивідуального акаунта користувача – підходить для більшості проектів, суть полягає в тому що абсолютно будь-який користувач може створити свій акаунт, також дозволяє авторизовувати користувачів за допомогою сторонніх сервісів, таких як Google, Twitter тощо;
- Аутентифікація для організацій – підходить для компаній, організацій, коли необхідно видавати доступ за корпоративною поштою;
- Windows аутентифікація – система аутентифікації для мереж Intranet за допомогою акаунтів Windows.

При розробці даного веб-додатку було використано перший спосіб, а саме аутентифікація індивідуального акаунта користувача. Також для реалізації було б можливо і навіть доцільно використовувати другий спосіб – аутентифікацію для користувачів, в цьому випадку школа виступала б організацією та могла б мати власний домен електронної пошти та були б створені облікові записи для кожного школяра. Таке рішення б допомогло уникнути проблеми з реєстрацією користувачів, які не мають відношення до даної школи. В даному випадку цю проблему було вирішено наступним чином: системним-адміністратором створюються індивідуальні логіни, котрі прив’язуються до учнів/вчителів.

Для неавторизованих користувачів з даного модулю демонструється представлення Login.cshtml. Дане представлення містить форму авторизації, а також посилання на перехід для форми реєстрації, яка реалізована у представленні Register.cshtml. Користувачі, які отримали свій логін мають

змогу зареєструватися на сторінці за допомогою цієї форми. Для цього потрібно ввести отриманий логін та створити пароль.

В ASP.NET Identity існує вже вбудована система захисту пароля. Дана система побудована на використанні хеш-ключів. З представлення Register.cshtml введений пароль, за допомогою AJAX-запиту, передається на контролер AccountController (метод Register), де за допомогою спеціальних хеш-функцій пароль перетворюється в хеш-ключ та зберігається в такому вигляді у базі даних. Функції зворотного перетворення з хеш-ключа в пароль не існує, тому навіть, якщо зловмисники отримають доступ до бази даних, вони не зможуть негайно отримати паролі.

В ситуації, коли користувач вже зареєстрований та має власний акаунт, він має можливість ввести свої дані для входу (логін та пароль) у формі авторизації (представлення Login.cshtml). Після вводу дані авторизації у вигляді моделі LoginViewModel передаються з представлення Login.cshtml на контролер AccountController (метод Login), де перевіряється наявність доступу у користувача, а саме пароль перетворюється в хеш-ключ, після цього отриманий хеш-ключ порівнюється з хеш-ключем, який прив’язаний до логіну даного користувача. Також за допомогою запиту до бази даних визнається роль користувача: вчитель/учень. В разі співпадіння хеш-ключів користувач вважається авторизованим та отримує доступ до внутрішніх можливостей веб-додатку.

Для авторизованого користувача даний модуль виглядає як веб-сторінка зміни паролю. Даній сторінці відповідає представлення ResetPassword.cshtml. Для зміни паролю необхідно вводити старий пароль, а також новий. Ця інформація у вигляді моделі ResetPasswordViewModel передається на контролер AccountController (метод ResetPassword). В даному методі відбувається перевірка старого паролю, тільки у разі його відповідності відбувається оновлення старий хеш-ключ в базі даних замінюється новим, отриманим з нового паролю.

Наступний модуль відповідає виставлення та перегляд оцінок. Даний модуль складається з наступних файлів:

- MarksController.cs;
- MarksTeacherPage.cshtml;
- MarksStudentPage.cshtml.

MarksController.cs – контролер, який відповідає за обробку HTTP-запитів, що поступають до даного модулю. Має кілька методів, які діляться на дві групи, в залежності від ролі користувача, одна група реалізує логіку потрібну для користувачів з роллю «вчитель», інша група – з роллю «учень».

Для користувачів з роллю «вчитель» демонструється HTML-розмітка, згенерована з допомогою представлення MarksTeacherPage.cshtml. Дане представлення містить таблицю з оцінками, яку можливо редагувати, а також фільтри для вибору проміжку часу, предмета та класу.

Для користувачів з роллю «учень» демонструється HTML-розмітка, згенерована з допомогою представлення MarksStudentPage.cshtml. Веб-сторінка, згенерована з використання даного представлення, містить фільтри для вибору часового проміжку та предмету, а також саму таблицю з онцінками.

3.4. Опис інтерфейсу веб-додатку

Розглянемо детальніше інтерфейс веб-додатку, який було розроблено в даному проєкті.

Роботу даного веб-додатку було протестовано у веб-браузерах: Google Chrome та Opera. З даними браузера додаток є повністю сумісним, крім того даний додаток є кросплатформеним, тобто його робота не залежить від операційної системи користувача.

При переході на веб-сторінку даного додатку неавторизований користувач бачить форму авторизації (рис 3.4). Для входу необхідно ввести свій

логін та пароль та натиснути кнопку «Вхід». У разі вводу некоректного паролю і/або логіну користувачу буде видано повідомлення про помилку авторизації. У разі, якщо користувач ще не створив свій пароль, необхідно перейти за посиланням на «Реєстрація нового користувача» .

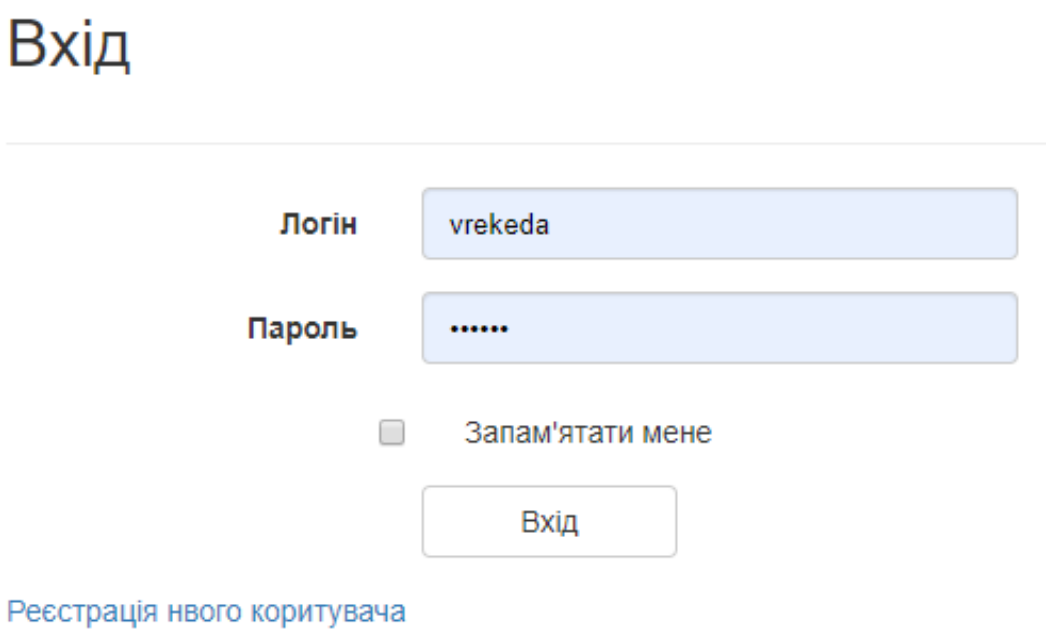

Рисунок 3.4 – Форма авторизації

Після переходу за раніше згаданим посиланням відкривається форма реєстрації (рис 3.5). Дана форма містить наступні поля:

- Логін;
- Пароль;
- Підтвердження паролю.

У полі «Логін» необхідно ввести той логін, який було створено шкільним системним адміністратором, даний логін є унікальним. В полі «Пароль» необхідно ввести власноруч придуманий пароль, задля покращення захисту пароль має містити не менше 8 символів, серед яких має бути хоча б одна цифра та хоча б одна маленька латинська літера, в паролі допускається використання великих латинських літер, а також небуквенних символів. У разі, якщо введений пароль не буде відповідати цим вимогам, користувач отримає відповідне повідомлення. У полі «Підтвердження паролю» необхідно ще раз

ввести обраний пароль. Після цього потрібно натиснути кнопку «Реєстрація». У разі, якщо паролі не співпадають або введений логін не було створено адміністратором, користувач отримає повідомлення про помилку, якщо все гаразд, буде здійснено перехід на особисту сторінку.

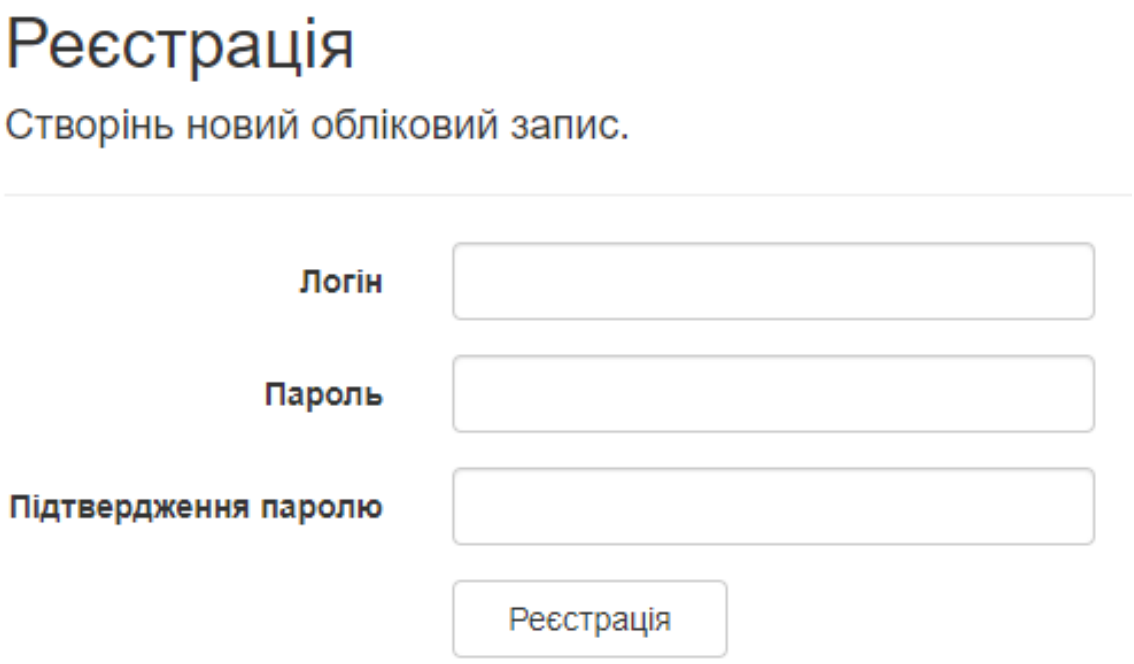

Рисунок 3.5 – Форма реєстрації

Після авторизації та переходу в особистий кабінет користувачу з роллю «Вчитель» будуть доступні наступні сторінки:

- Домашня сторінка;
- Оцінки;
- Сторінка налаштування облікового запису.

Перехід між даними сторінками доступний за допомогою навігаційної панелі у верхній частині веб-сторінки.

Розглянемо сторінку «Оцінки»(рис. 3.6).

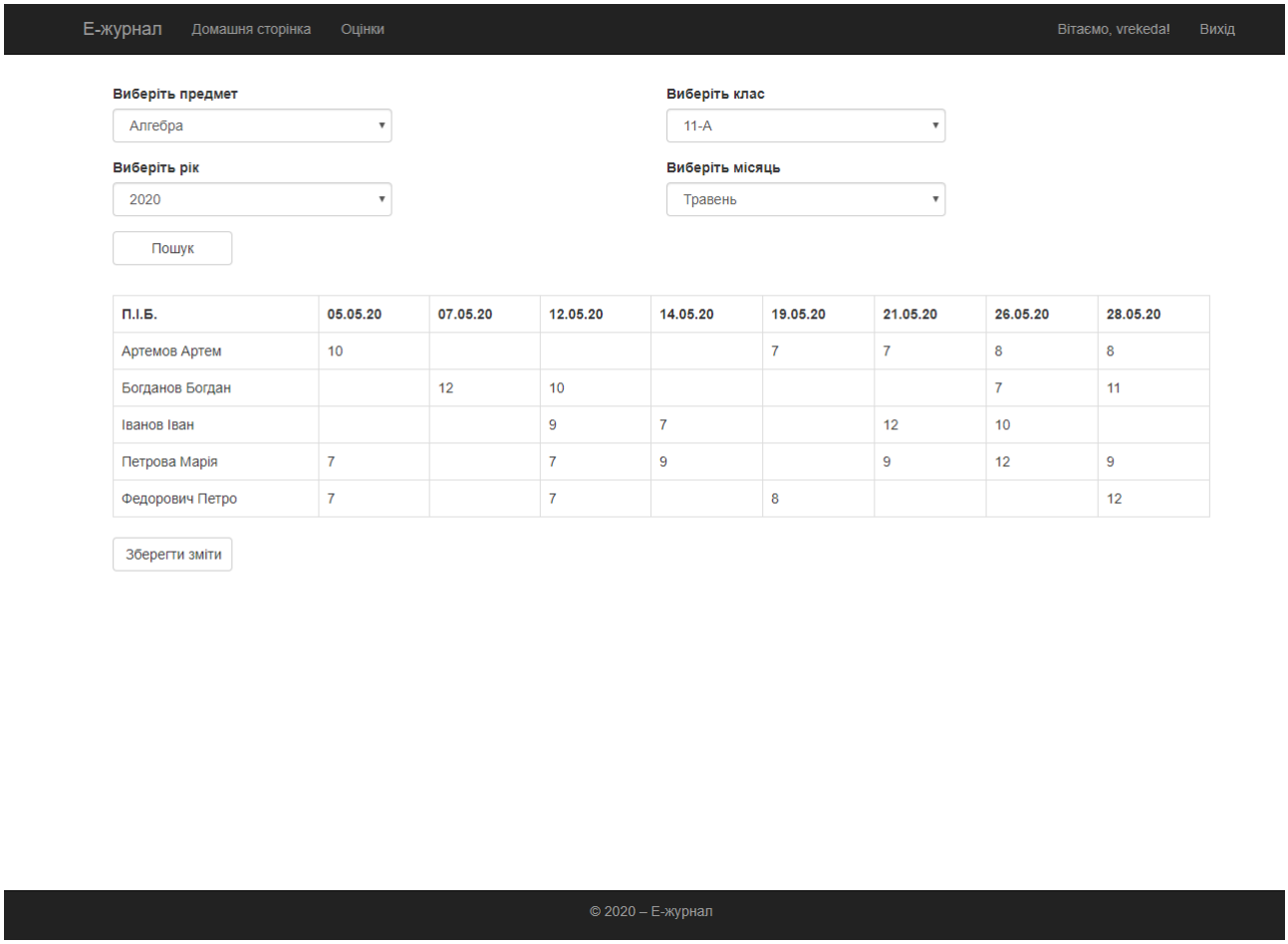

Рисунок 3.6 – Сторінка заповнення оцінок

У верхній частині сторінки знаходяться фільтри пошуку для вибору класу, предмету, року та місяця. Після того як фільтр було налаштовано, необхідно натиснути кнопку «Пошук». Після цього буде відображено таблицю з учнями вибраного класу та їх оцінками по обраному період за обраний проміжок часу. Дану таблицю можна вільно редагувати, після внесення всіх змін необхідно натиснути кнопку «Зберегти зміни».

3.5. Рекомендації щодо майбутнього покращення веб-додатку

Так як даний додаток було виконано на базі клієнт-серверної архітектури та з розбиттям програми на окремі модулі. Таке розбиття дозволяє досить

просте впровадження нових модулів та покращення функціоналу вже існуючих модулів.

Розглянемо можливі покращення веб-додатку. Одним з таких є покращення безпеки за рахунок додавання двофакторної ідентифікації. Система ідентифікації ASP.NET Identity підтримує дану функцію. Наприклад, можна використовувати електронну пошту для надсилання коду підтвердження, але для цього необхідно створити власний SMTP-сервер для надсилання повідомлень.

Задля збільшення функціоналу, який надається розробленим додатком, можна збільшити функціонал модуля оцінювання. Наприклад, можливо додати вид контролю, за який було виставлено оцінку, тобто чи це була контрольна робота, самостійна робота, домашня робота тощо.

Також є можливим впровадження модулю формування звітності. Так як всі необхідні дані знаходяться в БД, на серверній частині необхідно лише створити процедури обробки цих даних, а на частині клієнта необхідно реалізувати вивід таких звітів, крім того також можливо додати функцію вивантаження цих звітів у наступних форматах XLS, XLSX, CSV та деяких інших.

3.6. Висновки

Даний розділ містить опис процесу реалізації веб-додатку для ведення журналу успішності школярів, а саме було описано інструменти, які використовувалися при розробці даного програмного продукту. До цих інструментів відносяться мови програмування, що використовувалися для написання тексту програми. Крім того, було описано задіяні фреймворки, бібліотеки та середовище розробки.

Також було описано основні модулі розробленої програми, систему авторизації. Також була згадана інтерфейсна частина веб-орієнтованого додатку, з якою буде взаємодіяти користувач, а саме основні елементи, які є тими, що використовуються найчастіше.

Завдяки тому, що при розробці веб-додатку було використано шаблон проєктування MVC, покращення вже існуючого та додавання ного функціоналу є досить простим. Поради, щодо можливих покращень були описані в даному розділі.

ВИСНОВКИ

В ході роботи над даним дипломним проєктом було досліджено архітектурні особливості веб-орієнтованих додатків, а також проаналізовано технології, які використовуються при розробці таких додатків. Було створено веб-додаток для ведення журналу успішності школярів.

У першому розділі проєкту було проаналізовано актуальність проблеми, а саме необхідність створення подібного веб-додатку, описано вже існуючі подібні рішення. Аналіз показав, що проблема є нині актуальною, адже існування подібного додатку може допомогти в спрощенні комунікації між вчителями, учнями та їх батьками. Дослідження, що подібні програмні засоби частково вирішують дану проблему, але вони неповністю підходять під задані вимоги.

В другому розділі дипломного проєкту було описано особливості розробки веб-орієнтованих додатків, а саме: принцип роботи, архітектурні особливості. Також в даному розділі було розглянуто кілька популярних шаблонів проєктування, проведено їх порівняння, описано недоліки та переваги кожного з них. Серед даних шаблонів було обрано один, який і було використано при розробці веб-додатку. Також було розглянуто види СКБД, їх відмінності та особливості, обрану одну з них для використання у веб-додатку.

Третій розділ було присвячено процесу розробки веб-додатку, а саме описано інструменти, з допомогою яких, було розроблено веб-додаток, модулі веб-додатку, принцип їх роботи, а також інтерфейс. Крім того, було описано можливі варіанти покращення створеного веб-додатку.

Поставлену задачу було виконано цілком успішно. Розроблений веб-додаток для ведення журналу успішності школярів готовий до використання, а також має потенціал для покращення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Безкоштовна платформа створення шкільного сайту e-schools - [Електронний ресурс]. – Режим доступу: https://e-schools.info/. – Дата доступу: квітень 2020.
2. Українська освітня мережа Щоденник.ua - [Електронний ресурс]. – Режим доступу: http://company.shodennik.ua/. – Дата доступу: квітень 2020.
3. Web Application Architecture: Principles, Protocols and Practices [Текст]: підручник / Leon Shklar, Rich Rosen. – John Wiley and Sons, Ltd, 2017. – 178 с. – Дата доступу: квітень 2020.
4. WebSocket - Інтерфейси веб API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org /uk/docs/Web/API/WebSocket. – Дата доступу: квітень 2020.
5. The DCI Architecture: A New Vision of Object-Oriented Programming [Електронний ресурс]. – Режим доступу: https://www.artima.com/articles/dci_vision.html. – Дата доступу: травень 2020
6. Professional ASP.NET MVC 5 [Текст]: підручник Jon Galloway, Brad Wilson, K. Scott Allen, David Matson, 2014. – 132 ст. – Дата доступу: квітень 2020.
7. GUI Architectures [Електронний ресурс]. – Режим доступу: https://www.martinfowler.com/eaaDev/uiArchs.html. – Дата доступу: травень 2020.
8. Architecture Patterns: Model-View-ViewModel WebSocket [Електронний ресурс]. - Режим доступу: https://medium.com/upday-devs/android-architecture-patterns-part-3-model-view-viewmodel-e7eeee76b73b. – Дата доступу: травень 2020.

9. What a Relational Database Is [Електронний ресурс]. – Режим доступу: https://www.oracle.com/database/what-is-a-relational-database/. - Дата доступу: травень 2020.
10. Razor syntax reference for ASP.NET Core [Електронний ресурс]. – 2018. – Режим доступу: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/views/razor. – Дата доступу: квітень 2020.
11. C# IntelliSense [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-csharp-intellisense/. – Дата доступу: квітень 2020.